JP9200000304

#2 Petition 2/19/02 C. McKinley

J1021 U.S. PTO 10/023147 12/18/01

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２０００年１２月１９日

出願番号
Application Number: 特願２０００－３８６０６９

出願人
Applicant(s): インターナショナル・ビジネス・マシーンズ・コーポレーション

２００１年　６月１９日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３０５７７６４

【書類名】　特許願

【整理番号】　JP9000304

【提出日】　平成12年12月19日

【あて先】　特許庁長官殿

【国際特許分類】　G06F　9/30

【発明者】

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内

【氏名】　高野　光司

【発明者】

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内

【氏名】　佐藤　証

【特許出願人】

【識別番号】　390009531

【氏名又は名称】　インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】　100086243

【弁理士】

【氏名又は名称】　坂口　博

【代理人】

【識別番号】　100091568

【弁理士】

【氏名又は名称】　市位　嘉宏

【代理人】

【識別番号】　100106699

【弁理士】

【氏名又は名称】　渡部　弘道

【復代理人】

【識別番号】　100112520

【弁理士】

【氏名又は名称】　林　茂則

【選任した復代理人】

【識別番号】　100110607

【弁理士】

【氏名又は名称】　間山　進也

【手数料の表示】

【予納台帳番号】　091156

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9706050

【包括委任状番号】　9704733

【包括委任状番号】　0004480

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　演算回路および演算方法

【特許請求の範囲】

【請求項１】　複数のレジスタと、前記複数のレジスタに入力される値を入力とする演算器と、複数のメモリと、を含み、

前記複数のメモリから前記複数のレジスタへの複数の変数の読出しを、前記演算器のパイプライン処理における同一の読出しステージで行う演算回路。

【請求項２】　前記演算器は、第１レジスタ、第２レジスタ、第３レジスタ、第４レジスタの各々に入力されたｒビット長を有するx_1、x_2、x_3、x_4の各入力値に基づいて、２ｒまたは２ｒ＋１ビット長を有する$x_1+x_2\cdot x_3+x_4$の演算結果Ｑを与える積和演算器である請求項１記載の演算回路。

【請求項３】　前記複数のメモリには、第１メモリおよび第２メモリを含み、

前記パイプライン処理の演算ステージに続く演算結果の書き込みステージにおいて、前記演算結果Ｑの下位ｒビットQ_Lが前記第１メモリに記録され、前記演算結果Ｑの前記Q_Lを除く上位ビットQ_Hが前記第４レジスタに入力され、

前記書き込みステージに続く前記レジスタへの変数の読出しステージにおいて、前記第１メモリから前記第１レジスタに変数x_1が、前記第２メモリから前記第３レジスタに変数x_3が、同一の読出しステージで読み出される請求項２記載の演算回路。

【請求項４】　前記第１および第２メモリは、データの書き込みポートと読出しポートとを各々１つ有する２ポートメモリである請求項３記載の演算回路。

【請求項５】　前記第１メモリはデータの書き込みポートと読出しポートとを各々１つ有する２ポートメモリであり、前記第２メモリはデータの書き込みおよび読出しが１つのポートで行われる１ポートメモリである請求項３記載の演算回路。

【請求項６】　前記演算器は、第１レジスタ、第２レジスタ、第３レジスタ、第４レジスタ、第５レジスタ、第６レジスタの各々に入力されたｒビット長を有するx_1、x_2、x_3、x_4、x_5、x_6の各入力値に基づいて、２ｒまたは

２ｒ＋１ビット長を有する$x_1+x_2\cdot x_3+x_4\cdot x_5+x_6$の演算結果Ｑを与える積和演算器である請求項１記載の演算回路。

【請求項７】　前記複数のメモリには、第１メモリ、第２メモリおよび第３メモリを含み、

前記パイプライン処理の演算ステージに続く演算結果の書き込みステージにおいて、前記演算結果Ｑの下位ｒビットQ_Lが前記第１メモリに記録され、前記演算結果Ｑの前記Q_Lを除く上位ビットQ_Hが前記第６レジスタに入力され、

前記書き込みステージに続く前記レジスタへの変数の読出しステージにおいて、前記第１メモリから前記第１レジスタに変数x_1が、前記第２メモリから前記第３レジスタに変数x_3が、前記第３メモリから前記第５レジスタに変数x_5が、同一の読出しステージで読み出される請求項６記載の演算回路。

【請求項８】　前記第１メモリはデータの書き込みポートと読出しポートとを各々１つ有する２ポートメモリであり、前記第２および第３メモリは、データの書き込みおよび読出しが１つのポートで行われる１ポートメモリである請求項７記載の演算回路。

【請求項９】　複数の入力レジスタを有する演算器と複数のメモリとを備えた演算回路を用いた演算方法であって、

前記入力レジスタに入力されている値に基づいて演算を行うステップと、

前記演算の結果を前記入力レジスタまたは前記メモリに書き込むステップと、

前記複数のメモリから前記複数の入力レジスタに複数の変数を同一のパイプラインステージで読み出すステップと、

を含む演算方法。

【請求項１０】　前記演算器は、第１レジスタ、第２レジスタ、第３レジスタ、第４レジスタの各々に入力されたｒビット長を有するx_1、x_2、x_3、x_4の各入力値に基づいて、２ｒまたは２ｒ＋１ビット長を有する$x_1+x_2\cdot x_3+x_4$の演算結果Ｑを与える積和演算器である請求項９記載の演算方法。

【請求項１１】　前記複数のメモリには、第１メモリおよび第２メモリを含み、

前記演算結果Ｑの下位ｒビットQ_Lが前記第１メモリに記録され、前記演算結

果Ｑの前記$Ｑ_{L}$を除く上位ビット$Ｑ_{H}$が前記第４レジスタに入力される前記演算器のパイプライン処理における書き込みステップと、

前記第１メモリから前記第１レジスタへの変数$ｘ_{1}$の読出しと、前記第２メモリから前記第３レジスタへの変数$ｘ_{3}$の読出しとが前記パイプライン処理の同一の読出しステージで行われる読出しステップと、

を含む請求項１０記載の演算方法。

【請求項１２】　前記第１および第２メモリは、データの書き込みポートと読出しポートとを各々１つ有する２ポートメモリである請求項１１記載の演算方法。

【請求項１３】　前記第１メモリはデータの書き込みポートと読出しポートとを各々１つ有する２ポートメモリであり、前記第２メモリはデータの書き込みおよび読出しが１つのポートで行われる１ポートメモリである請求項１１記載の演算方法。

【請求項１４】　前記演算器は、第１レジスタ、第２レジスタ、第３レジスタ、第４レジスタ、第５レジスタ、第６レジスタの各々に入力されたｒビット長を有する$ｘ_{1}$、$ｘ_{2}$、$ｘ_{3}$、$ｘ_{4}$、$ｘ_{5}$、$ｘ_{6}$の各入力値に基づいて、２ｒまたは２ｒ＋１ビット長を有する$ｘ_{1}＋ｘ_{2}\cdot ｘ_{3}＋ｘ_{4}\cdot ｘ_{5}＋ｘ_{6}$の演算結果Ｑを与える積和演算器である請求項９記載の演算回路。

【請求項１５】　前記複数のメモリには、第１メモリ、第２メモリおよび第３メモリを含み、

前記演算結果Ｑの下位ｒビット$Ｑ_{L}$が前記第１メモリに記録され、前記演算結果Ｑの前記$Ｑ_{L}$を除く上位ビット$Ｑ_{H}$が前記第６レジスタに入力される前記演算器のパイプライン処理における書き込みステップと、

前記第１メモリから前記第１レジスタへの変数$ｘ_{1}$の読出しと、前記第２メモリから前記第３レジスタへの変数$ｘ_{3}$の読出しと、前記第３メモリから前記第５レジスタへの変数$ｘ_{5}$の読出しとが、前記パイプライン処理の同一の読出しステージで行われる読出しステップと、

を含む請求項１４記載の演算方法。

【請求項１６】　前記第１メモリはデータの書き込みポートと読出しポート

とを各々１つ有する２ポートメモリであり、前記第２および第３メモリはデータの書き込みおよび読出しが１つのポートで行われる１ポートメモリである請求項１５記載の演算方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、演算回路および演算方法に関し、特に公開鍵暗号化方式等に好適な乗剰余計算の高速化に関するものである。

【０００２】

【従来の技術】

情報を伝送する際に、セキュリティの確保あるいは認証のために公開鍵暗号（非対称暗号）が用いられる。公開鍵暗号は公開鍵と秘密鍵の一対の鍵を用いて情報を伝送する暗号方式である。送信者が受信者の公開鍵で平文を暗号化し、暗号文を受取った受信者は受信者しか知りえない秘密鍵を用いて暗号文を復号する。このような公開鍵暗号では共通鍵暗号（対称鍵暗号）のように１つの共通鍵を共有する必要がなく、また、公開鍵暗号では公開鍵を広く公開することができるので不特定多数の者との通信の秘密が確保できる。さらに、公開鍵暗号を電子認証あるいは電子署名に用いて面識のない他人との信頼関係を確立することができる。インターネット等通信技術が支えるネットワーク社会あるいはその中で営まれる商取引等において必須の技術と位置付けられる。

【０００３】

公開鍵暗号方式の一つにＲＳＡが知られている。ＲＳＡは非常に大きな整数の離散対数問題あるいは素因数分解の困難性にその安全性の基礎をおく。たとえば、公開鍵（ｅ，ｎ）を用いて、平文Ｍを

$$C = M^{e} \ (\mathrm{mod}\ n)$$

の関係式から暗号文Ｃを生成する（なお、Ｍは整数ｎ未満になるようブロック化されている）。この暗号文Ｃの解読には離散対数計算（ａ、ｙ、ｐからｙ＝ａ^{x}（ｍｏｄ　ｐ）となるｘを見出す）が必要となり

$$O(2^{SQRT(\log n)})$$

の計算量を必要とする（ＳＱＲＴは二乗根を与える関数である）。整数ｎが少なくとも５１２ビット長以上、好ましくは１０２４ビット長以上であれば実用的な計算時間での解読は困難になる。

【０００４】

ところが、公開鍵（ｅ，ｎ）と

ｅｄ（ｍｏｄ　ｌｃｍ（ｐ－１、ｑ－１））＝１　，　ｎ＝ｐｑ、（ただしｐ、ｑは十分大きな素数）

の関係にある秘密鍵（ｄ，ｎ）を用いれば、

$M=C^{d}$（ｍｏｄ　ｎ）

の関係式を用いて平文Ｍを簡単に求めることができる（ただし、ｌｃｍ（ａ，ｂ）はａとｂの最小公倍数を与える）。

【０００５】

たとえば、ｄ＝１１の場合、

$C^{11}=((C^{2})^{2}C)^{2})C$

のようにｄを２進展開して、自乗剰余演算と乗算剰余演算を繰り返せば、高々ｄのビット長の２倍の回数の乗剰余演算で計算できる。

【０００６】

しかしながら、上記べき乗剰余演算であっても、たとえばＤＥＳ（ｄａｔａ　ｅｎｃｒｙｐｔｉｏｎ　ｓｔａｎｄａｒｄ）等の対称暗号に比較すれば計算量が多くなる。このためできるだけ効率的なアルゴリズムとその実装が求められる。

【０００７】

上記べき乗剰余演算における自乗剰余演算と乗算剰余演算を高速化する手法に、たとえばPeter L. Montgomery著、「Modular Multiplication Without Trial Division」、Mathematics of computations, Vol.44, No.170 April 1985, pp 519-522、に記載されているモンゴメリ乗算手法がある。モンゴメリ乗算は乗剰余算を、加算、乗算、シフト演算の繰り返しにより、減算を繰り返す除算よりも少ない計算量で実現する手法である。以下にモンゴメリ乗算の計算主要部

$P\equiv XYR^{-1}$（ｍｏｄ　ｎ）

を擬似コード１．ｘに示す。なお、上式において、

$R=(2^{r})^{m}$

$N\equiv -n^{-1} \pmod{2^{r}}$

とする。また、擬似コードにおいて各行の左側には行番号を付す（以下同様）。

【０００８】

```
(1.1)    P = 0;
(1.2)    for (i=0 ; i < m ; i++){
(1.3)         t = (p0 + xiy0) N (mod 2^r);
(1.4)         P = (P + xiY + t·n) / 2^r;
(1.5)    };
(1.6)    if (P => n) P = P-n;
```

上記擬似コード１．ｘに示すように、その主要部分の繰り返し演算は次のようになる。まず、Ｘをｍ個のブロックx_iに区切り（$X=(x_{m-1}, x_{m-2}, \cdots, x_1, x_0)$）、Ｙとの部分積加算（$x_i Y$）をｍ回繰り返す（行番号１．２～１．５）。このとき、途中結果Ｐの最下位ブロックp_0が０となるようなｎの倍数「ｔ・ｎ」を毎回加算する（行番号１．４）。ｔは行番号１．３において定義されている。さらに、Ｐをｒビットだけ右にシフト、つまり「2^{-r}」を乗ずる（行番号１．４）。なお、ｒビットのシフト演算は、ｍ回のシフト演算により$2^{-rm}=R^{-1}$となってR^{-1}の乗算をすることになる。

【０００９】

たとえば、５１２ビットのモンゴメリ乗算を３２ビット乗算器によって実行すると仮定すると、ｍ＝５１２／３２＝１６回のループを繰り返すことになる。上記擬似コードでは簡単のためにx_i・Ｙや、ｔ・ｎのように３２ビット×５１２ビットのように示したが、実際は５１２ビットのＹとｎも１６個の３２ビットブロックに分割して計算を実行する。つまり、Ｐの部分積加算は演算においてｍ＝１６の二重ループとなる。以下に二重ループによってモンゴメリ乗算を実行する手順の一例を擬似コード２．ｘに示す。

【００１０】

```
(2.1)    P = 0;
(2.2)    for (i=0 ; i < m ; i++){
```

```
(2.3)        t = p_0 + x_i y_0 (mod 2^r);
(2.4)        t = t·N (mod 2^r);
(2.5)        c = 0;
(2.6)        for (j=0 ; j < m ; j++){
(2.7)             tmp = p_j + x_i·y_j + c;
(2.8)             tmp = tmp + t·n_j;
(2.9)             if (j != 0) p_j-1 = tmp(mod 2^r);
(2.10)            c = tmp/2^r;
(2.11)       };
(2.12)       p_m-1 = c;
(2.13)  };
(2.14)  if (P => n) P = P-n;
```

なお、ここで、Ｘ，Ｙ，ｎはｍ個のブロックに分割されている。つまり、

$X=(x_{m-1}, x_{m-2}, \cdots, x_1, x_0)$

$Y=(y_{m-1}, y_{m-2}, \cdots, y_1, y_0)$

$n=(n_{m-1}, n_{m-2}, \cdots, n_1, n_0)$

乗算器が一つの場合を仮定すると、途中結果ｔｍｐの計算に二回の積和演算が必要となる。変数p_j，x_i，y_j，ｔ，n_jは全てｒビット長の数、変数ｃは下位ブロックからのキャリーである。上記擬似コード２．ｘの例では、一回のｊループで２ｒビットの数$x_i \cdot y_j$とｔ・n_j、そしてｒ＋１ビットの数p_jとｃの加算を行い（行番号２．６～２．１１）、積和演算後の途中結果ｔｍｐは２ｒ＋１ビット長を持つ。ｔｍｐの下位ｒビットは変数p_jに，上位ｒ＋１ビットは変数ｃにストアされる（行番号２．９、２．１０）。

【００１１】

一方、$x_i \cdot y_j$とｔ・n_jの加算を２つの別のループで行うことも可能である。この例を擬似コード３．ｘに示す。

【００１２】

```
(3.1)    P = 0;
(3.2)    for (i=0 ; i < m ; i++){
```

```
(3.3)        c = 0;
(3.4)        for (j=0 ; j < m ; j++){
(3.5)             tmp = p_j + x_i·y_j + c;
(3.6)             p_j = tmp(mod 2^r);
(3.7)             c = tmp/2^r;
(3.8)        };
(3.9)        p_m = c; c = 0;
(3.10)       t = p_0·N (mod 2^r);
(3.11)       for (j=0 ; j < m ; j++){
(3.12)            tmp = p_j + t·n_j + c;
(3.13)            if (j != 0) p_j-1 = tmp(mod 2^r);
(3.14)            c = tmp/2^r;
(3.15)       };
(3.16)       p_m-1 = p_m + c;
(3.17)  };
(3.18)  if (P => n) P = P-n;
```

擬似コード３．ｘの例では、変数ｐ$_j$はｒビット長、変数ｔｍｐは２ｒビット長となる。

【００１３】

擬似コード２．ｘ、３．ｘの両例ともモンゴメリ乗算の二重ループを抜けた後の結果Ｐは２ｎ未満となるが、ｎ以下である保証はないので必要に応じて、「Ｐ＝Ｐ－ｎ」とする（行番号２．１４，３．１８）。

【００１４】

上記擬似コード２．ｘ、３．ｘの例において、基本的にｐ$_j$は、Ｐをｍ個に分割したｒビットレジスタで正の値をとる。しかし、擬似コード２．ｘの例では、行番号２．１２の「ｐ$_{m-1}$＝ｃ」において、ｃは最大ｒ＋１ビットとなる。この時、演算ビット数がｒビットで割り切れる場合には１ビットあふれることになる。よって、ループ回数をｍ＋１にするか、ループ回数をｍ回に押さえるための最上位ビットの特別な処理が必要となる。本明細書では簡単のため上記例の両者

ともｊループは同じループ回数ｍであるものとする。なお両例の行番号２．１４，３．１８の比較演算「ｉｆ　（Ｐ＝＞ｎ）」において、符号ビットの考慮などを行う必要がある。

【００１５】

【発明が解決しようとする課題】

上記擬似コード２．ｘ、３．ｘの例により、モンゴメリ乗算の計算を実行することができる。しかし、本発明者らがその演算サイクルを詳細に検討したところ、演算のパフォーマンスは変数のレジスタ割り付けやメモリ構成に大きく影響されることが判明した。以下検討結果について説明する。

【００１６】

上記擬似コード２．ｘの場合、ｒビット長の変数Ｎはｉループ（擬似コード２．２～２．１３）内で値が変わらずに繰り返し使用される。また、ｔとx_iはｊループ（擬似コード２．６～２．１１）内で値が変らずに繰り返し使用される。このため、変数Ｎ，ｔ，x_iについては、一旦計算あるいはメモリから読み出した後は各ループを抜けるまでレジスタに保持して繰り返し使用することができる。またｊループ内で使用する途中変数ｔｍｐとｃはレジスタに直接割り付けられる。その他の変数は，読み出しと書き込みが独立したアドレスに対して実行できる一般的な２ポートメモリに記録する。またメモリ読み出し→演算→書き込みの一連の処理はパイプライン処理されるものとする。そうすると、演算と書き込みは基本的に１サイクルで終了するのでメモリ読み出しがパフォーマンスのボトルネックになる。以下、メモリ読み出しのサイクル数を検討する。

【００１７】

まず行番号２．１の「Ｐ＝０」の処理については、行番号２．７の演算

$$tmp = p_j + x_i \cdot y_j + c$$

などで最初にp_jを読み出すときにその値をリセットすればよい。このためサイクル数０である。

【００１８】

行番号２．３，２．４の変数ｔの計算

$$t = p_0 + x_i y_0 \pmod{2^r}$$

$t = t \cdot N \pmod{2^r}$

ではレジスタに割り付けられているＮ以外の変数（p_0、x_i、y_0）をメモリから読み出すのに３サイクルを要する。なお、ｔについては前記した通り行番号２．３の演算結果が直接レジスタに割り付けられるので読出しの必要はない。行番号２．５の「ｃ＝０」は、「Ｐ＝０」の処理と同様０サイクルである。

【００１９】

ｊループに入って、行番号２．７の演算

$tmp = p_j + x_i \cdot y_j + c$

はp_jとy_jの読み出しで２サイクルを必要とする。x_iは行番号２．３のｔの計算の際に読み出されているのでここでの読出しの必要はない。また、ｃについては前記の通り直接レジスタに割り付けられるので読出しの必要はない。

【００２０】

行番号２．８の演算

$tmp = tmp + t \cdot n_j$

はn_jの読み出しで１サイクルを要する。ｔｍｐ、ｔについては前記した通り直接レジスタに割り付けられるので読出しの必要はない。

【００２１】

行番号２．９の演算

$p_{j-1} = tmp \pmod{2^r}$

$c = tmp / 2^r$

の実際のハードウェア上の動作は、「$tmp + t \cdot n_j$」を一旦レジスタｔｍｐで受けずに直接p_{j-1}とｃに書き込めばよい。よってどちらも０サイクルとなる。

【００２２】

ｊループを抜けた直後の行番号２．１２の演算

$p_{m-1} = c$

はパイプライン処理により０サイクルとなる。

【００２３】

したがってｉループ内の演算サイクル数は、

３＋ｍ（２＋１）＝３ｍ＋３

となる。これにｉループのｍを乗じて

ｍ（３ｍ＋３）＝３ｍ^{2}＋３ｍ

となる。しかし、先に述べたようにｉ＝０のときｐ$_{j}$＝０なのでｐ$_{j}$を読み出す必要はなく、入力を０リセットすればよい。よってｉループ全体ではこの分のｍを減じて

３ｍ^{2}＋３ｍ－ｍ＝３ｍ^{2}＋２ｍ

となる。行番号２．１４の比較動作

ｉｆ　（Ｐ　＝＞　ｎ）

では減算「Ｐ－ｎ」によって大小を判定するのでこれに２ｍサイクル、最後にパイプラインを抜けるのに２サイクルを要する。結局最終的なサイクル数は

３ｍ^{2}＋４ｍ＋２

となる。これはパイプライン処理が乱れなく実行できる場合であり、演算ビット数が乗算器のビット数に対してあまり大きくないとき、つまりブロック数ｍが少ないときは、書き込もうとした値を直ぐに読み出す必要があるためメモリアクセス待ちのオーバーヘッドが加わることになる。ただしブロック数が４～５あれば実用上このような問題は生じない。また逆にブロック数が２～３といった小さな数に対してモンゴメリ法を適用する意味はあまりないので、上記事情は考慮する必要がない。つまり、二重ループによるｍ^{2}項の係数「３」が大きく変わることはない。また、実装形態によってはｊループ前後でパイプライン動作が乱れることがあり、前記最終的なサイクル数が実際には若干異なることがある。しかし、この場合であっても二重ループによるｍ^{2}項の係数「３」は変わらず、最終的なサイクル数が前記値と大きく異なることはない。

【００２４】

これと同様に擬似コード３．ｘの例を詳細に検討すれば、以下の通りである。行番号３．１の「Ｐ＝０」の処理は、行番号３．５の演算

ｔｍｐ＝ｐ$_{j}$＋ｘ$_{i}$・ｙ$_{j}$＋ｃ

で最初にｐ$_{j}$を読み出すときにその値を０リセットすればよいのでサイクル数は０である。行番号３．１の「ｃ＝０」の処理についても、擬似コード２．ｘと同

様０サイクルである。

【００２５】

第１ｊループ（行番号３．４～３．８）に入って、行番号３．５の演算

$tmp = p_j + x_i \cdot y_j + c$

では、p_jとy_jの読み出しで２サイクルを必要とし、第１ｊループの最初にx_iを読み出す必要がある。行番号３．６の演算

$p_j = tmp \pmod{2^r}$

$c = tmp / 2^r$

は擬似コード２．ｘの場合と同様にｔｍｐを直接p_j、ｃに書き込むので０サイクルとなる。よって、１番目のｊループのサイクル数は、２ｍ＋１となる。

【００２６】

第１ｊループを抜けた後の行番号３．９の演算

$p_m = c;\quad c = 0;$

はパイプライン処理により０サイクルとなり、行番号３．１０の演算

$t = p_0 \cdot N \pmod{2^r}$

では、p_0とＮの読み出しに２サイクルを要する。

【００２７】

次に第２ｊループ（行番号３．１１～３．１５）に入って、行番号３．１２の演算

$tmp = p_j + t \cdot n_j + c$

では、p_jとn_jの読み出しで２サイクルを要する。なお、ｔとｃはレジスタに直接書込まれるので読出しの必要はない。それに続く行番号３．１３の演算

$p_{j-1} = tmp \pmod{2^r}$

$c = tmp / 2^r$

は、擬似コード２．ｘの場合と同様０サイクルとなる。従って、２番目のｊループのサイクル数は２ｍとなる。そして２番目のｊループを抜けた後の行番号３．１６の演算

$p_{m-1} = p_m + c$

では、p_mの読出しに１サイクル必要である。

【００２８】

よって、ｉループ内の演算サイクル数は、

（２ｍ＋１）＋２＋２ｍ＋１＝４ｍ＋４

となる。これにｉループのｍを乗じて

ｍ（４ｍ＋４）＝４ｍ^{2}＋４ｍ

となる。しかし、先に述べたようにｉ＝０のときｐ$_{j}$を読み出す必要はなく、入力を０リセットすればよいので、ｉループ全体ではこの分のｍを減じて

４ｍ^{2}＋４ｍ－ｍ＝４ｍ^{2}＋３ｍ

となる。行番号３．１８の比較動作

ｉｆ　（Ｐ＝＞ｎ）

では減算「Ｐ－ｎ」によって大小を判定するのでこれに２ｍサイクル、最後にパイプラインを抜けるのに２サイクルを要する。結局最終的なサイクル数は

４ｍ^{2}＋５ｍ＋２

となる。擬似コード３．ｘの例でｍ^{2}サイクル多いのは、ｔｍｐへの部分積加算を一つのｊループで行わないため、行番号３．４～３．８のループでメモリに書き込んだｐ$_{j}$を行番号３．１１から３．１５のループで読み出すためである。サイクル数こそ多いものの桁上がりが１ビット少なくｒビットブロックにおさまり、レジスタに保持すべき変数も少ないという擬似コード２．ｘの例と比較したメリットがある。このため、制御と回路規模の点では３．ｘの例の方が有利である。図７に擬似コード３．ｘの場合のモンゴメリ乗算回路データパス部の概略を、また図８にこの回路におけるブロック数ｍ＝４の最終ループ処理近辺のタイミングチャートを示す。なお、パイプライン処理の乱れ等により、前記最終的な読出しサイクル数が実際には若干変化することがあるのは擬似コード２．ｘの場合と同様である。

【００２９】

上記のメモリ読出しタイミングにおいてモンゴメリ乗算の高速化を考えるとするなら、二重ループの内側で乗算を２回行うので、単純に乗算器を２つ実装すれば２倍の速度が得られるように思える。しかし、メモリ読み出しがボトルネックとなるのは前記した通りである。このボトルネック解消の方法として、独立な読

み出しを行える読出しポートを２つ、書き込みポートを１つ持つ３ポートメモリを使用することが考えられる。しかしながら３ポートメモリにより高速化が図れるものの、３ポートメモリは一般的ではなく、使用できるテクノロジーが限定される。すなわち、回路の汎用性が損なわれてしまう。このため半導体設計における回路設計の標準化（ＩＰコア化）の障害になる可能性がある。さらに、３ポートメモリは素子面積が大きくなり小型化の要請にそぐわない。

【００３０】

本発明の目的は、モンゴメリ乗算回路におけるメモリアクセスのボトルネックを２ポートやシングルポートの汎用メモリを使いながら解消することにある。これにより高いパフォーマンスと素子の小型化を実現し、暗号生成回路を低いコストで提供することが可能になる。

【００３１】

【課題を解決するための手段】

本発明の概要を説明すれば以下の通りである。すなわち、本発明の演算回路および演算方法では、１サイクルで実行する演算に必要な変数のうちメモリから読み出す必要のある変数については異なるメモリに記録し、１つの読出しステージでこれら変数を同時に読み出すことを可能にしたものである。これにより、読出しによる待ち時間を最小にして計算のパフォーマンスを向上することができる。しかも、変数を記録するメモリには３ポートメモリ等複雑な回路、大面積を要する回路を用いることなく、２ポートやシングルポートの汎用的なメモリを用いてＩＰコア化をし易くし、また素子面積を最小限にすることを可能にする。

【００３２】

以下本発明を列記すれば以下の通りである。すなわち、本発明の演算回路は、複数のレジスタと、前記複数のレジスタに入力される値を入力とする演算器と、複数のメモリとを有し、前記複数のメモリから前記複数のレジスタへの複数の変数の読出しを、前記演算器のパイプライン処理における同一の読出しステージで行うものである。また、本発明の演算方法は、複数の入力レジスタを有する演算器と複数のメモリとを備えた演算回路を用いた演算方法であって、前記入力レジスタに入力されている値に基づいて演算を行うステップと、前記演算の結果を前

記入力レジスタまたは前記メモリに書き込むステップと、前記複数のメモリから前記複数の入力レジスタに複数の変数を同一のパイプラインステージで読み出すステップと、を含む。

【００３３】

なお、前記演算器は、第１レジスタ、第２レジスタ、第３レジスタ、第４レジスタの各々に入力されたｒビット長を有するx_1、x_2、x_3、x_4の各入力値に基づいて、２ｒまたは２ｒ＋１ビット長を有する$x_1+x_2\cdot x_3+x_4$の演算結果Ｑを与える積和演算器とすることができる。この場合、前記複数のメモリには、第１メモリおよび第２メモリを含み、前記パイプライン処理の演算ステージに続く演算結果の書き込みステージにおいて、前記演算結果Ｑの下位ｒビットQ_Lが前記第１メモリに記録され、前記演算結果Ｑの前記Q_Lを除く上位ビットQ_Hが前記第４レジスタに入力され、前記書き込みステージに続く前記レジスタへの変数の読出しステージにおいて、前記第１メモリから前記第１レジスタに変数x_1が、前記第２メモリから前記第３レジスタに変数x_3が、同一の読出しステージで読み出されるものとすることができる。また、前記第１および第２メモリを、データの書き込みポートと読出しポートとを各々１つ有する２ポートメモリとすること、あるいは、前記第１メモリをデータの書き込みポートと読出しポートとを各々１つ有する２ポートメモリ、前記第２メモリをデータの書き込みおよび読出しが１つのポートで行われる１ポートメモリとすることができる。

【００３４】

また、前記演算器は、第１レジスタ、第２レジスタ、第３レジスタ、第４レジスタ、第５レジスタ、第６レジスタの各々に入力されたｒビット長を有するx_1、x_2、x_3、x_4、x_5、x_6の各入力値に基づいて、２ｒまたは２ｒ＋１ビット長を有する$x_1+x_2\cdot x_3+x_4\cdot x_5+x_6$の演算結果Ｑを与える積和演算器とすることができる。この場合、前記複数のメモリには、第１メモリ、第２メモリおよび第３メモリを含み、前記パイプライン処理の演算ステージに続く演算結果の書き込みステージにおいて、前記演算結果Ｑの下位ｒビットQ_Lが前記第１メモリに記録され、前記演算結果Ｑの前記Q_Lを除く上位ビットQ_Hが前記第６レジスタに入力され、前記書き込みステージに続く前記レジスタへの変数の

読出しステージにおいて、前記第１メモリから前記第１レジスタに変数x_1が、前記第２メモリから前記第３レジスタに変数x_3が、前記第３メモリから前記第５レジスタに変数x_5が、同一の読出しステージで読み出されるものとすることができる。また。前記第１メモリをデータの書き込みポートと読出しポートとを各々１つ有する２ポートメモリとし、前記第２および第３メモリを、データの書き込みおよび読出しが１つのポートで行われる１ポートメモリとすることができる。

【００３５】

【発明の実施の形態】

以下、本発明の実施の形態を図面に基づいて詳細に説明する。ただし、本発明は多くの異なる態様で実施することが可能であり、本実施の形態の記載内容に限定して解釈すべきではない。なお、実施の形態の全体を通して同じ要素には同じ番号を付するものとする。

【００３６】

（実施の形態１）

図１は、本発明の一実施の形態であるモンゴメリ乗算回路の一例をそのデータパス部について示したブロック図である。本実施の形態のモンゴメリ乗算回路は積和演算回路１と、入力レジスタ２～５と、マルチプレクサ６と、２つのメモリ７（メモリＡ）およびメモリ８（メモリＢ）とを有する。

【００３７】

積和演算回路１は、レジスタ３とレジスタ４の入力値（ｒビット長）の積に、レジスタ２とレジスタ５の入力値（ｒビット長）を加算して２ｒビット長の出力ｔｍｐを与える。すなわち、加算入力p_j，ｃと積算入力y_j，x_iとから出力$tmp = p_j + y_j \cdot x_i + c$を得る。積和演算回路１は、たとえば公知の全加算器ＦＡと半加算器ＨＡとを用いて構成できる。なお、図１において$x_{i(0)}$のように示しているのは、変数x_iの括弧内の数字に相当するビットを抜き出した値に対応する。

【００３８】

入力レジスタ２～５には、メモリ７，８から値が読み出されて入力され、ある

いは積和演算回路１からの出力が直接入力される。マルチプレクサ６は３つの入力に対し１つの出力を与えるスイッチである。

【００３９】

メモリ７，８は汎用的な２ポートメモリ、すなわちデータの書き込みと読出しを各々独立した１つのポートから行えるメモリである。メモリ７（メモリＡ）には二重ループ処理中の途中変数Ｐ（p_j）が記録され、メモリ８（メモリＢ）には、その他の変数Ｘ（x_i），Ｙ（y_j），ｎ（n_j），Ｎと最終結果Ｐ（p_j）が記録される。メモリ７，８は何れも２ポートメモリであり、素子のＩＰコア化の障害、素子面積の増大の問題を生じない。

【００４０】

本実施の形態の演算回路では、データの読出し、演算、書き込みが一連のパイプラインとして処理される。すなわち、メモリからレジスタへのデータの読出しステージと、レジスタに入力された値に基づく演算ステージと、演算結果をレジスタまたはメモリに書き込む書き込みステージとが並列に処理される。

【００４１】

図２は、図１の回路において擬似コード３．ｘの処理を実行した時の処理のタイミングを示したタイミングチャートである。図２では、ｍ＝４の場合の処理の最終ループ近辺を示している。なお前記および以下のタイミングチャートにおいて一連のパイプライン処理（データの読出し、演算、結果の書き込み）を同じ背景色で示すようにグレーまたは白に濃淡分けてして示している。また、チャートの左端には、読出し先のメモリ（MemA(read)、MemB(read)）、レジスタ（p_j（レジスタ５）、y_j，n_j，Ｎ（レジスタ４）、ｃ（レジスタ２）、x_i，ｔ（レジスタ３））、書込み先のメモリ（MemA(write)、MemB(write)）を指標として示している。チャートの各行には、読出しステージにおいて各メモリから読み出される値、演算ステージにおけるレジスタ内の値、書込みステージにおけるメモリ書込まれる値が記入されている。

【００４２】

擬似コード３．ｘの第１ｊループ（行番号３．４～３．８）における処理、たとえば図２のｉ＝３ループ中の最初のｊ＝０～３の処理では、まず、ｊ＝０のル

ープで、メモリ７（ＭｅｍＡ）からレジスタ５（p_j）にp_0を、メモリ８（ＭｅｍＢ）からレジスタ４（y_j，n_j，Ｎ）にy_0を読み出す（読出しステージ）。なおこの読出しステージと同時に前サイクルの演算ステージ（行番号３．１６の計算）および前々サイクルの書込みステージ（メモリ７（ＭｅｍＡ）へのp_2の書込み、ｃレジスタへの上位ビットtmp_Hの書込み）が行われている。

【００４３】

次に、ｊ＝１の読出しステージでメモリ７（ＭｅｍＡ）からレジスタ５（p_j）にp_1を、メモリ８（ＭｅｍＢ）からレジスタ４（y_j，n_j，Ｎ）にy_1を読み出すと同時に先に読み込まれたp_0、y_0とレジスタ２（ｃ）に入力されている前ステップの演算結果（tmp_H）およびレジスタ３（x_i，ｔ）に入力されているx_3に基づいて行番号３．５の演算を実行する（演算ステージ）。なお、この時前サイクルの書込みステージ（メモリ７（ＭｅｍＡ）へのp_3の書込み、ｃレジスタ（レジスタ２）への上位ビットtmp_Hの書込み）が行われている。

【００４４】

次に、ｊ＝２の読出しステージでメモリ７（ＭｅｍＡ）からレジスタ５（p_j）にp_2を、メモリ８（ＭｅｍＢ）からレジスタ４（y_j，n_j，Ｎ）にy_2を読み出すと同時に前サイクルでレジスタに読み出された値に基づいて演算を実行し、先の演算結果（変数p_0、x_3、y_0、ｃに基づく演算結果）の上位ビットtmp_Hおよび下位ビットp_0を各々ｃレジスタとメモリ７（ＭｅｍＡ）に書き込む（書込みステージ）。このようにして、p_0、y_0の入力以降結果が記録されるまでの一連のパイプライン処理が行われる。その他のp_j、y_jについても同様である。

【００４５】

次に、行番号３．９の処理では、読み込みサイクルに１サイクルの空きを設け、ｊ＝３の演算ステージの後にレジスタｃの内容をp_4としてメモリ７に記録する。

【００４６】

次に、行番号３．１０の処理（ｔの計算）では、メモリ７からp_0を、メモリ

８からＮを同一の読出しステージで各々レジスタ３およびレジスタ４に読出し、演算の結果を次の書込みステージでレジスタ４に変数ｔとして書き込む。

【００４７】

次に、行番号３．１１～３．１５の第２ｊループの処理では、第１ｊループの場合と同様にメモリ７からp_jが、メモリ８からn_jが同一読出しステージで読み出され、先に入力された変数ｔ、ｃを用いて行番号３．１２の演算を行う。その後、演算結果の下位ビットはp_{j-1}としてメモリ７に記録され（行番号３．１３）、上位ビットはｃレジスタに記録される（行番号３．１４）。なお、この第２ｊループのｊ＝０における処理ではｃは０リセットされる。また、ｊ＝０では演算結果の下位ビットは記録されない。

【００４８】

行番号３．１６の処理では、メモリ７からp_4がレジスタ５（p_j）読み出され、ｃレジスタの値との和をとってp_3としてメモリ７に記録される。ただし、最後のｉループのメモリへの書込み処理では、p_jはメモリ７だけでなくメモリ８にも記録する。

【００４９】

最後に、行番号３．１８の処理では、メモリ７からp_jとn_jを各々レジスタに読出し、比較演算を行う。つまり減算「Ｐ＝Ｐ－ｎ」の結果はメモリ８（ＭｅｍＢ）の前回の結果Ｐを上書きしない場所に保存する。メモリ８（ＭｅｍＢ）は最後のｊループを抜けた時のＰと、それからｎを減じたＰ－ｎを保持することになるが、Ｐ－ｎが正であればＰ－ｎを、Ｐ－ｎが負であればＰを最終結果とする。最後のｊループを抜けた時にメモリ７（ＭｅｍＡ）にもＰを書き込むのは単にＰ－ｎの計算でメモリ８（ＭｅｍＢ）に保持されているｎと同時に読み出すためである。

【００５０】

上記のようにして、擬似コード３．ｘの演算処理を終了する。上記演算処理では、メモリからの変数の読出しを同一の読出しステージで行う。このように同一の読出しステージで変数の読出しが行えるのは、メモリを２つ設け、同時に読み出す可能性のある変数を相違するメモリに記録できるようにしたためである。こ

れにより、従来の方式に比較してメモリ読出しの待ち時間を短縮して、処理のパフォーマンスを向上できる。

【００５１】

上記処理による読出しステージのサイクル数を検討すれば、以下の通りである。第１ｊループの処理でｍ、行番号３．９，３．１０の処理で２、第２ｊループの処理でｍ、行番号３．１６の処理で１、よってｉループ内の処理に必要なサイクル数は２ｍ＋３となる。これにｉループのｍ回を乗じて$2m^2+3m$。なおｉ＝０のときにp_jを読み出す必要はないがそのサイクルでy_jを読み出す必要があるのでｍを減じることはない。行番号３．１８の比較動作にｍサイクル、パイプラインを抜けるのに２サイクルを要する。結局最終的なサイクル数は

$$2m^2+4m+2$$

となる。従来方式において$4m^2+5m+2$の読出しサイクルを必要としたことと比較すれば、約半分のサイクル数に減少する。

【００５２】

なお、前記実施の形態では擬似コード３．ｘの場合を説明したが、擬似コード２．ｘのように$x_i \cdot y_j$と$t \cdot n_j$を一つのループで加算することも可能である。擬似コード２．ｘの場合は、擬似コード３．ｘの場合に比べ変数p_jの読み出しの待ちサイクルが半分と少ないため、全体のサイクル数が少ないというメリットがあったが、前記実施の形態のようにメモリを２つ設けてこれを使い分ける場合、両者に違いはない。これは図２からもわかるように、本実施の形態ではp_jとy_jまたはp_jとn_jの読み出しを別のメモリから同時に行え、余計な待ちが生じないためである。本実施の形態では、擬似コード２．ｘの場合であっても、従来方式のサイクル数、$3m^2+4m+2$と比較して、そのおよそ２／３になる。p_{m-1}に余計な桁上がりが生じてｒ＋１ビットになることもないので、例外処理が不要であり、回路構成が簡単になるメリットもある。

【００５３】

（実施の形態２）

図３は、本発明の他の実施の形態であるモンゴメリ乗算回路の一例をそのデータパス部について示したブロック図である。本実施の形態の演算回路はメモリ８

（メモリＢ）が１ポートメモリ、つまりデータの読出しと書込みの双方を１つのポートで行うメモリとすることを除き、実施の形態１と同様である。このように本実施の形態ではメモリ８を１ポートメモリとすることにより、メモリサイズを小さくして回路の小型化を図ることが可能になる。

【００５４】

図４は、図３の回路において擬似コード３．ｘの処理を実行した時の処理のタイミングを示したタイミングチャートである。図４では、ｍ＝４の場合の処理の最終ループ近辺を示している。図示するようにｉループが終了するまでの処理は実施の形態１と同様である。Ｐ＝Ｐ－ｎの処理において、本実施の形態ではメモリ８に１ポートメモリを用いているので最終的な計算結果であるp_jの書込みをメモリ８に行わず、メモリ７に行う。そして、最後にメモリ７（ＭｅｍＡ）からメモリ８（ＭｅｍＢ）にp_jの転送を行う。すなわち、最終結果Ｐの補正演算は、Ｐ≧ｎならばＰ－ｎを最終結果とするものであるが、Ｐ≧ｎかどうかは減算を一回実行するまで判定できない。そのため実施の形態１ではＰとＰ－ｎの双方をメモリ８（ＭｅｍＢ）に書き込んで、減算結果の符合を見ていずれか一方を選択するようにした。本実施の形態ではＰとＰ－ｎをメモリ７（ＭｅｍＡ）にだけ書き込み、減算結果に応じてＰかＰ－ｎのいずれか一方をメモリ８（ＭｅｍＢ）に書き込むようにする。このようにすれば、ｍ＋１サイクル増加するだけのペナルティーでメモリ８（ＭｅｍＢ）をシングルポートメモリにすることができる。これによりメモリの小型化を図ることが可能になる。なお、本実施の形態の場合のサイクル数は

$2m^2+5m+3$

となる。これに対して、従来のよう１つのシングルポートメモリで擬似コード２．ｘを処理した場合、

$4m^2+5m+2$

擬似コード３．ｘを処理した場合、

$6m^2+7m+2$

のサイクル数が必要になる。本発明の優位性がより顕著となる。

【００５５】

（実施の形態３）

図５は、本発明のさらに他の実施の形態であるモンゴメリ乗算回路の一例をそのデータパス部について示したブロック図である。本実施の形態のモンゴメリ乗算回路は積和演算回路９と、入力レジスタ１０～１５と、３つのメモリ１６（メモリＡ）、メモリ１７（メモリＢ１）およびメモリ１８（メモリＢ２）とを有する。

【００５６】

積和演算回路９は、レジスタ１１（ｔレジスタ）とレジスタ１２（n_j，Ｎレジスタ）の入力値（ｒビット長）の積と、レジスタ１３（x_iレジスタ）とレジスタ１４（y_jレジスタ）の入力値（ｒビット長）の積と、レジスタ１０（ｃレジスタ）の入力値（ｒ＋１ビット長）と、レジスタ１５（p_jレジスタ）の入力値（ｒビット長）を加算した２ｒ＋１ビット長の出力ｔｍｐを与える。すなわち、積和演算回路９は２つの乗算器を有し、加算入力p_j，ｃと積算入力y_j，x_iと、積算入力ｔ，n_jとから出力$tmp = p_j + y_j \cdot x_i + t \cdot n_j + c$を得る。

【００５７】

レジスタについては実施の形態１と同様である。また、メモリ１６（ＭｅｍＡ）は汎用的な２ポートメモリであり、メモリ１７，１８（ＭｅｍＢ１，Ｂ２）はシングルポートメモリである。これら汎用的なメモリを用いるため、素子のＩＰコア化の障害が生じず、また、メモリ数の増加をシングルポートメモリを採用することによる素子面積の低減で補うことができる。

【００５８】

図６は、図５の回路において擬似コード２．ｘの処理を実行した時の処理のタイミングを示したタイミングチャートである。図５では、ｍ＝４の場合の処理の最終ループ近辺を示している。

【００５９】

まず、ｉループの最初にｔの計算（擬似コード２．３，２．４）を行う。メモリ１６からp_0を読出し、メモリ１７からx_3を読出し、メモリ１８からy_0を読み出す。これら読出しは１つの読出しステージで行われる。これら読み出した

変数を用いて行番号２．３の演算を行い、演算結果の下位ビットｔｍｐ$_L$をｔレジスタに入力する。前記演算と並行してメモリ１７からＮを読出し、前記ｔとＮを用いて行番号２．４の演算を行う。演算結果の下位ビットｔｍｐ$_L$がｔレジスタに入力され、本ループ内を通じてこのｔの値が用いられる。なお、先に読み出したｘ$_3$も本ループ内を通じて用いられる。

【００６０】

次にｊループに入り、ｐ$_0$，ｎ$_0$，ｙ$_0$が各々メモリ１６，１７，１８からｐ$_j$レジスタ、ｎ$_j$，Ｎレジスタ、ｙ$_j$レジスタに同一サイクル内で読み込まれる。その後、ｃ、ｔ、ｘ$_3$の値を用いて演算が行われ、演算結果の下位ビットはｐ$_{j-1}$としてメモリ１６に、上位ビットはｃレジスタに書込まれる（行番号２．９，２．１０）。なお、このｊループのｊ＝０における処理ではｃは０リセットされる。また、ｊ＝０では演算結果の下位ビットは記録されない。

【００６１】

行番号２．１４の処理では、ｃレジスタの値がｐ$_m$としてメモリ１６に記録される。その後のＰ＝Ｐ－ｎ、ＭｅｍＡ→ＭｅｍＢの処理は実施の形態２と同様である。

【００６２】

本実施の形態によれば、２つの乗算器を設け、さらにメモリを３つ設けたので、３変数を同時に読み出すことが可能になり、２つの乗算器を最大限に活用するデータの読出しを行うことが可能になる。

【００６３】

本実施の形態で必要な処理サイクルは以下の通りである。ｉループ内のｔの計算で２サイクル、ｊループでｍサイクル、擬似コード２．１２の処理で１サイクル、合計ｍ＋３サイクルがｉループ内の処理で必要である。ｉループのｍ回を乗じてｍ^2＋３ｍとなる。なおｉ＝０のときにｐ$_j$を読み出す必要はないがそのサイクルでｘ$_i$，ｙ$_j$を読み出す必要があるのでｍを減じることはない。行番号２．１４の比較動作とメモリ間転送に２ｍサイクル、パイプラインを抜けるのに２サイクルを要する。結局最終的なサイクル数は

ｍ^2＋５ｍ＋２

となる。従来方式はもとより、実施の形態１，２と比較しても大幅にサイクル数が低減される。特にｍの値が大きくなるとｍ^{2}項が効いてくるので、その効果がより顕著になる。本実施の形態を用いることにより２乗算器の場合のメモリアクセスのボトルネックを解消でき、サイクル数を大幅に改善することが可能になる。

【００６４】

以上、本発明者によってなされた発明を発明の実施の形態に基づき具体的に説明したが、本発明は前記実施の形態に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることは言うまでもない。たとえば、前記実施の形態では、メモリとしてシングルポートまたは２ポートのメモリを例示したが、３ポートメモリにも適用できる。

【００６５】

【発明の効果】

本願で開示される発明のうち、代表的なものによって得られる効果は、以下の通りである。すなわち、モンゴメリ乗算回路におけるメモリアクセスのボトルネックを２ポートやシングルポートの汎用メモリを使いながら解消することができる。これにより、高いパフォーマンスと素子の小型化を実現し、暗号生成回路を低いコストで提供できる。

【図面の簡単な説明】

【図１】

本発明の一実施の形態であるモンゴメリ乗算回路の一例をそのデータパス部について示したブロック図である。

【図２】

図１の回路において擬似コード３．ｘの処理を実行した時の処理のタイミングを示したタイミングチャートである。

【図３】

本発明の他の実施の形態であるモンゴメリ乗算回路の一例をそのデータパス部について示したブロック図である。

【図４】

図３の回路において擬似コード３．ｘの処理を実行した時の処理のタイミングを示したタイミングチャートである。

【図５】

本発明のさらに他の実施の形態であるモンゴメリ乗算回路の一例をそのデータパス部について示したブロック図である。

【図６】

図５の回路において擬似コード２．ｘの処理を実行した時の処理のタイミングを示したタイミングチャートである。

【図７】

従来方式において擬似コード３．ｘの場合のモンゴメリ乗算回路データパス部の概略を示すブロック図である。

【図８】

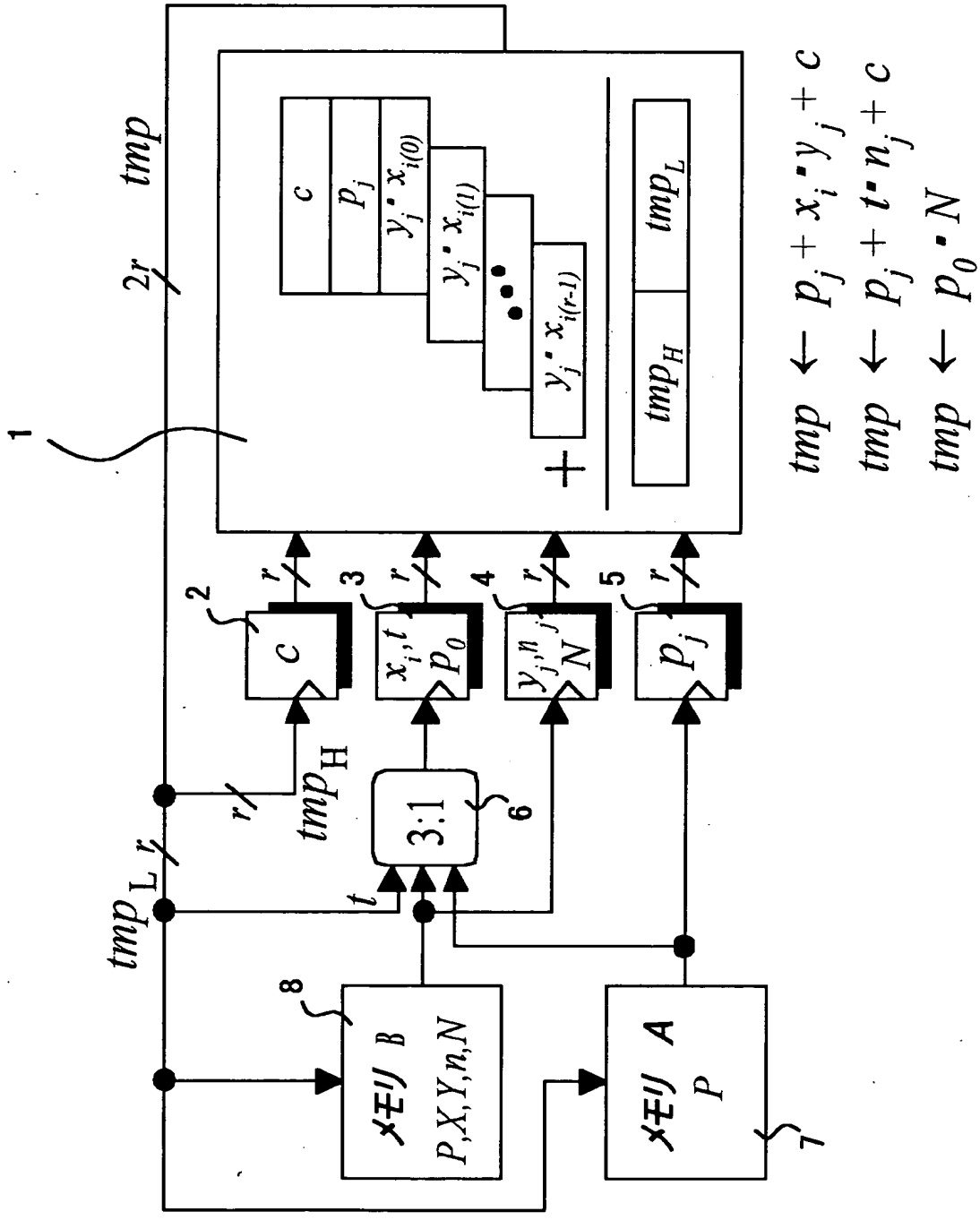
図７におけるブロック数ｍ＝４の最終ループ処理近辺のタイミングチャートである。

【符号の説明】

１，９…積和演算回路、２～５、１０～１５…入力レジスタ、６…マルチプレクサ、７，８，１６～１８…メモリ。

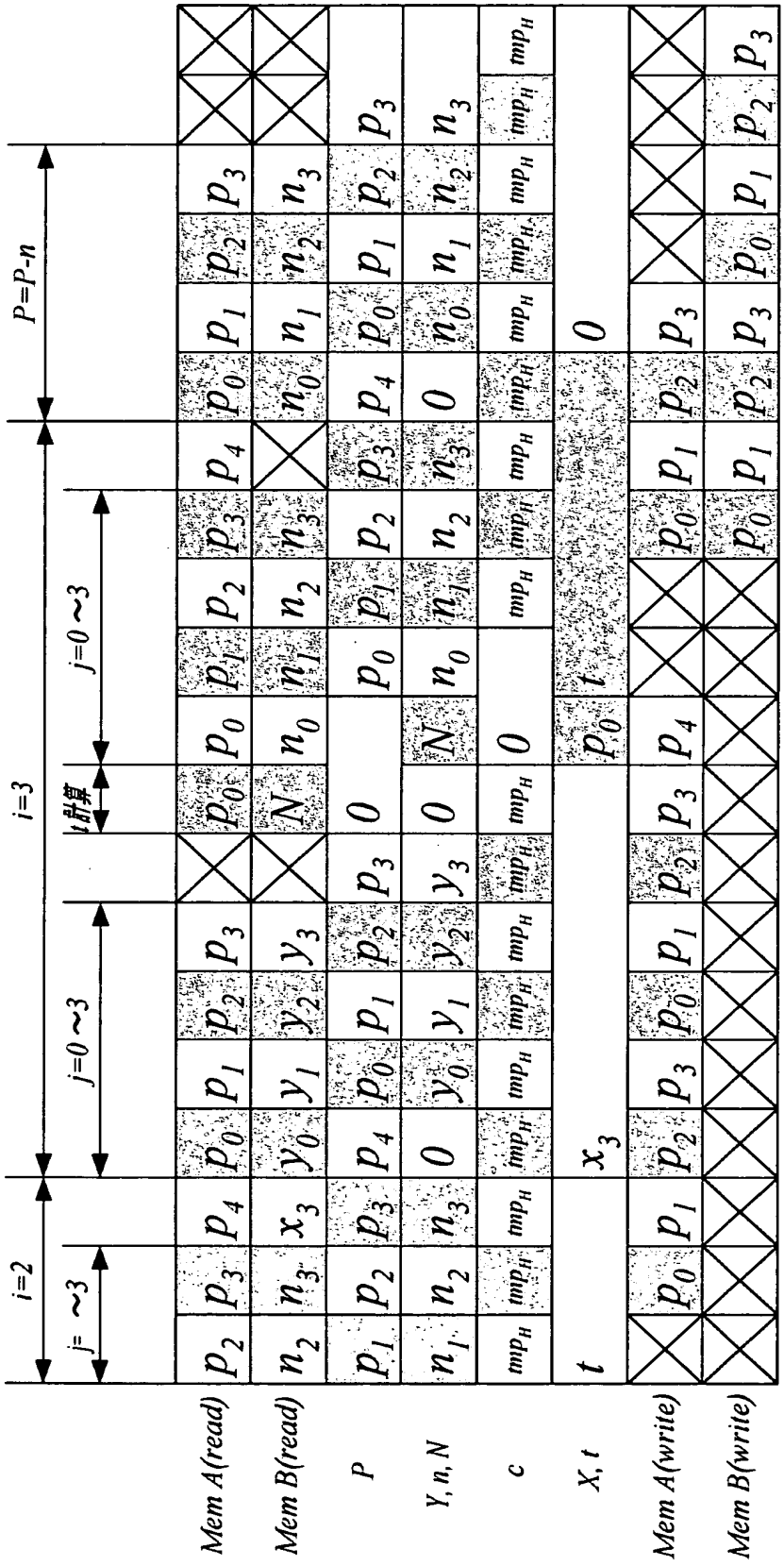
【書類名】　　図面

【図１】

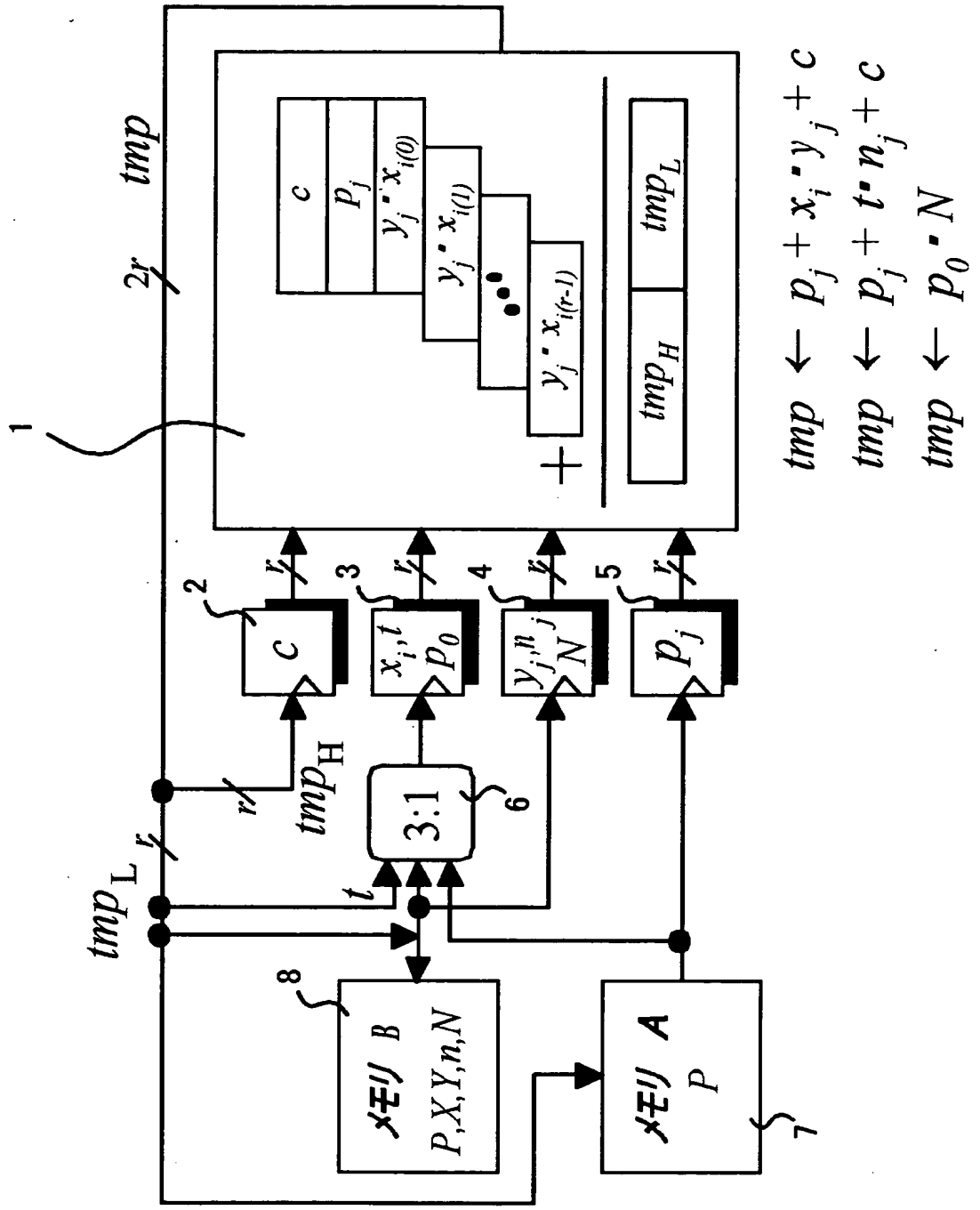


出証特２００１－３０５７７６４

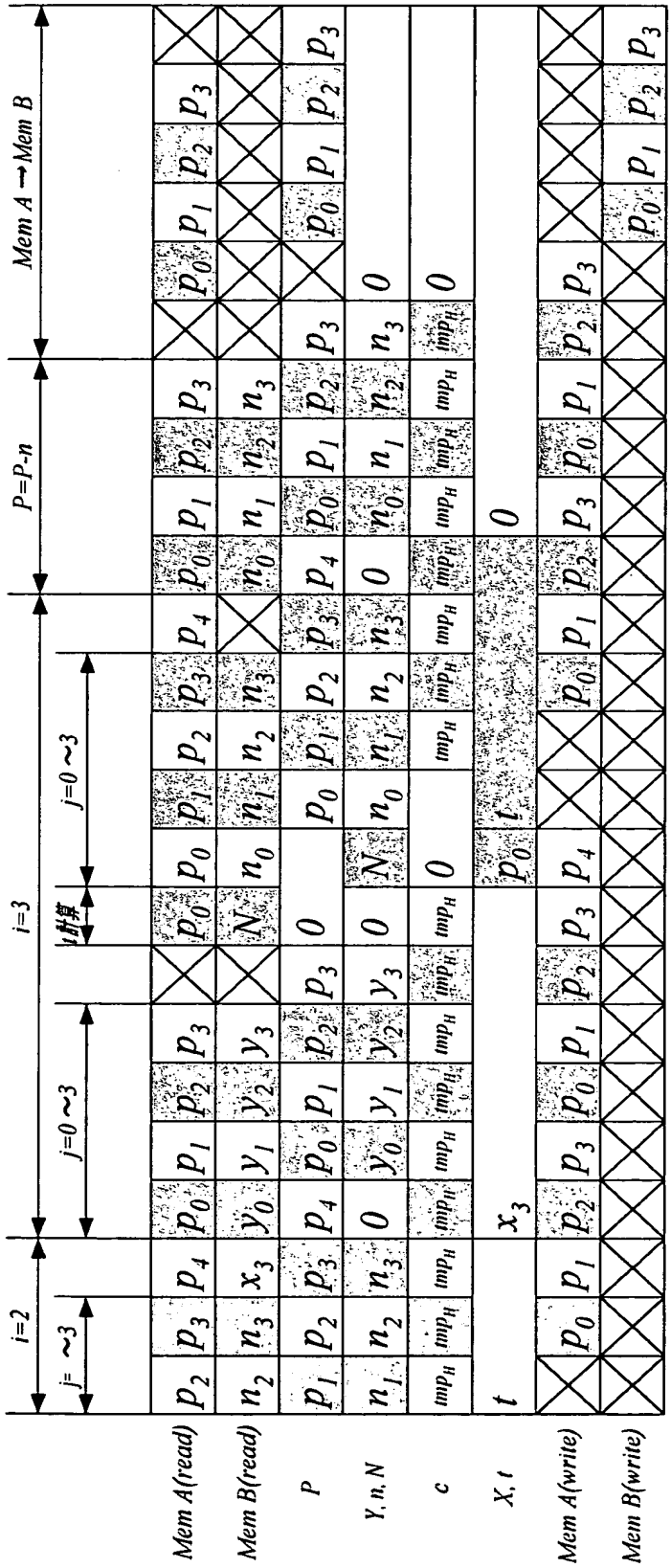
【図２】


i=2
j= ~3
j=0 ~3
i=3
t計算
j=0 ~3
P=P-n
Mem A(read)
Mem B(read)
P
Y, n, N
c
X, t
Mem A(write)
Mem B(write)

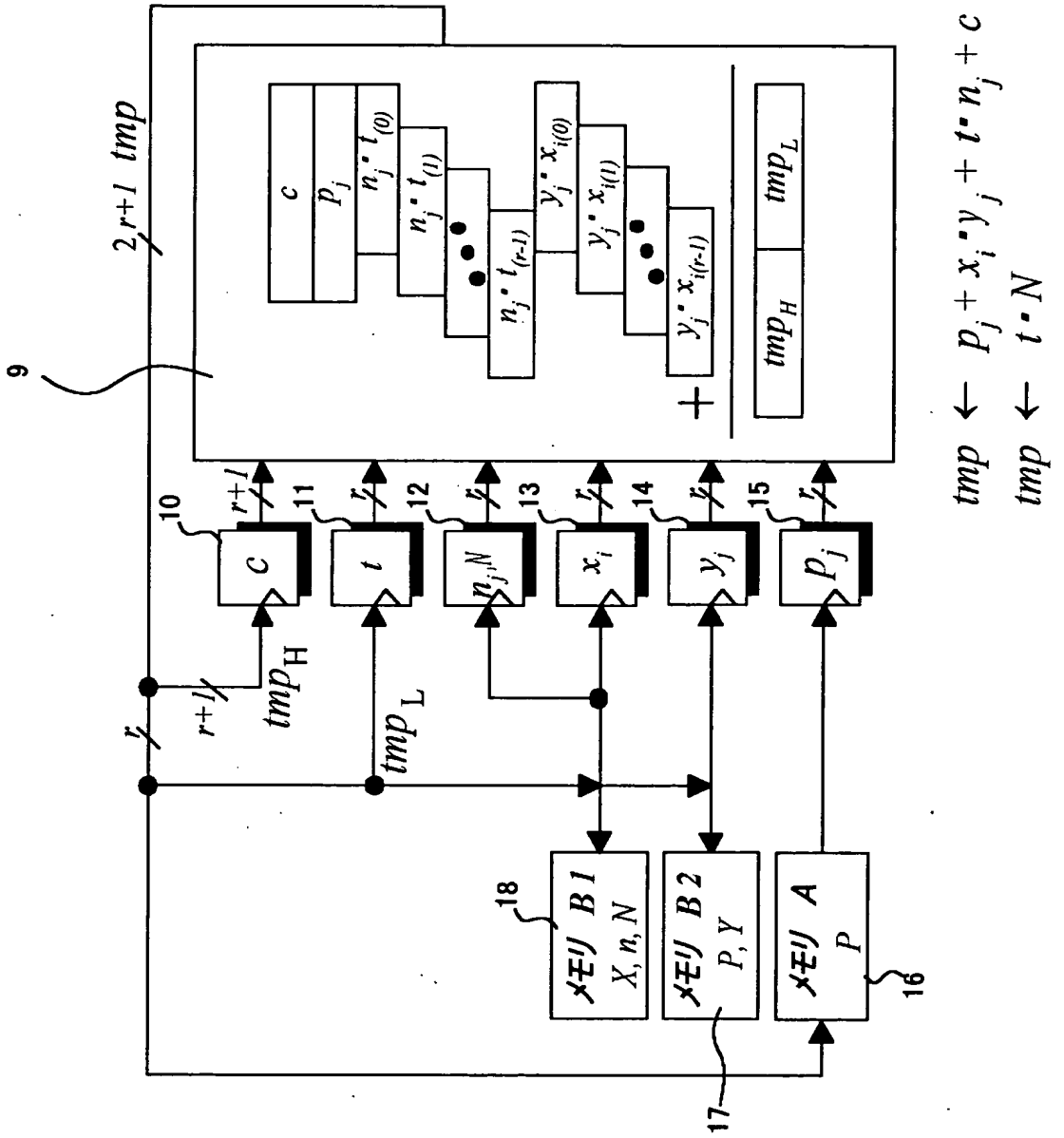
【図３】


tmp
2r
1
c
p_j
$y_j \cdot x_{i(0)}$
$y_j \cdot x_{i(1)}$
$y_j \cdot x_{i(r-1)}$
+
tmp_L
tmp_H
$tmp \leftarrow p_j + x_i \cdot y_j + c$
$tmp \leftarrow p_j + t \cdot n_j + c$
$tmp \leftarrow p_0 \cdot N$
2
3
4
5
r
c
x_i, t
p_0
y_j, n_j
N
p_j
tmp_H
tmp_L
3:1
6
t
8
メモリ B
P,X,Y,n,N
メモリ A
P
7

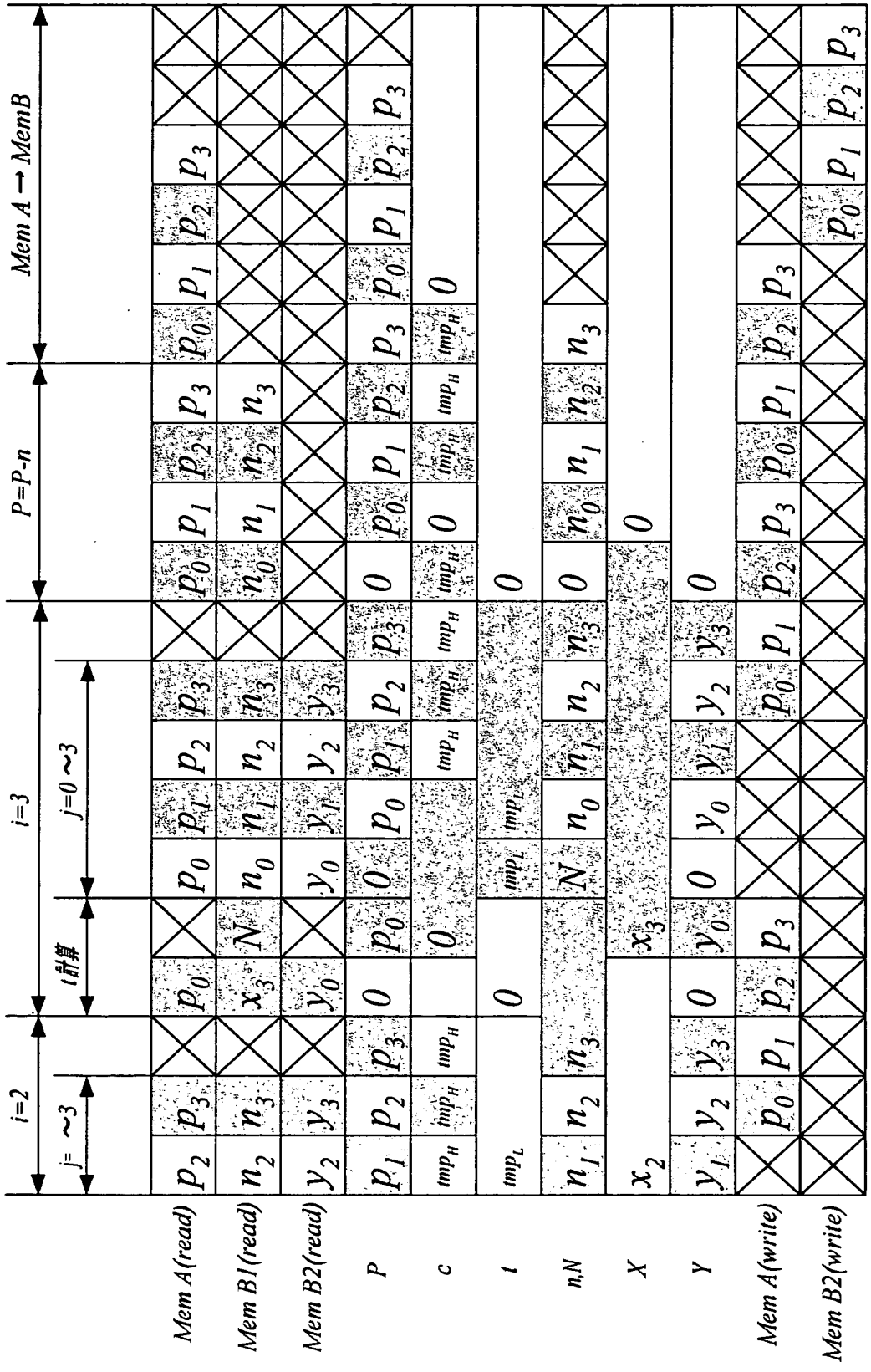
【図４】


i=2
j= ~3
j=0 ~3
i=3
j=0 ~3
P=P-n
Mem A → Mem B
Mem A(read)
Mem B(read)
P
Y, n, N
c
X, t
Mem A(write)
Mem B(write)

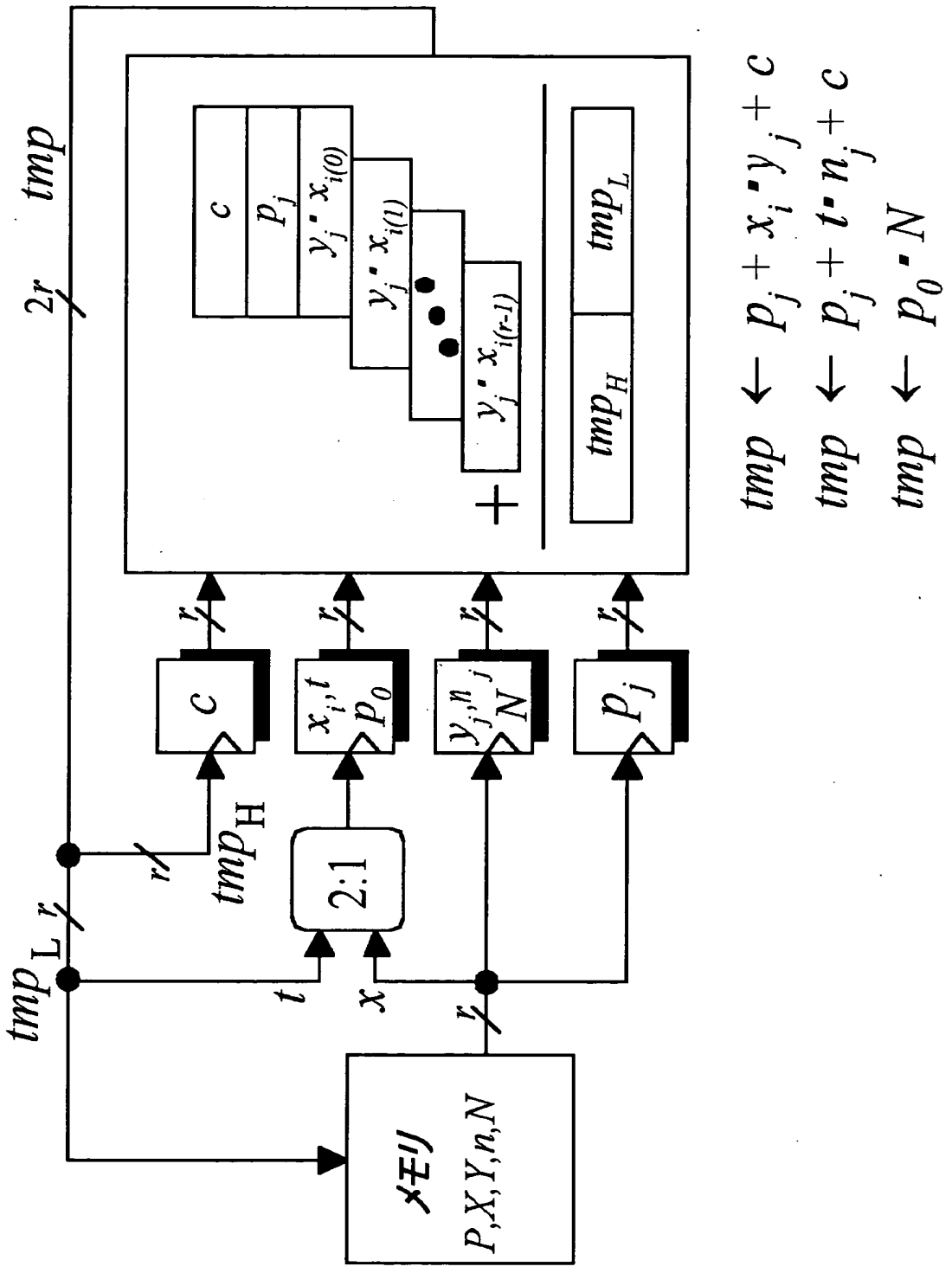
【図５】


9
2r+1 tmp
c
p_j
$n_j \cdot t_{(0)}$
$n_j \cdot t_{(1)}$
$n_j \cdot t_{(r-1)}$
$y_j \cdot x_{i(0)}$
$y_j \cdot x_{i(1)}$
$y_j \cdot x_{i(r-1)}$
+
tmp_L
tmp_H
r+1
r
10
11
12
13
14
15
c
t
$n_j N$
x_i
y_j
p_j
tmp_H
tmp_L
18
メモリ B1
X, n, N
メモリ B2
P, Y
メモリ A
P
16
17
$tmp \leftarrow p_j + x_i \cdot y_j + t \cdot n_j + c$
$tmp \leftarrow t \cdot N$

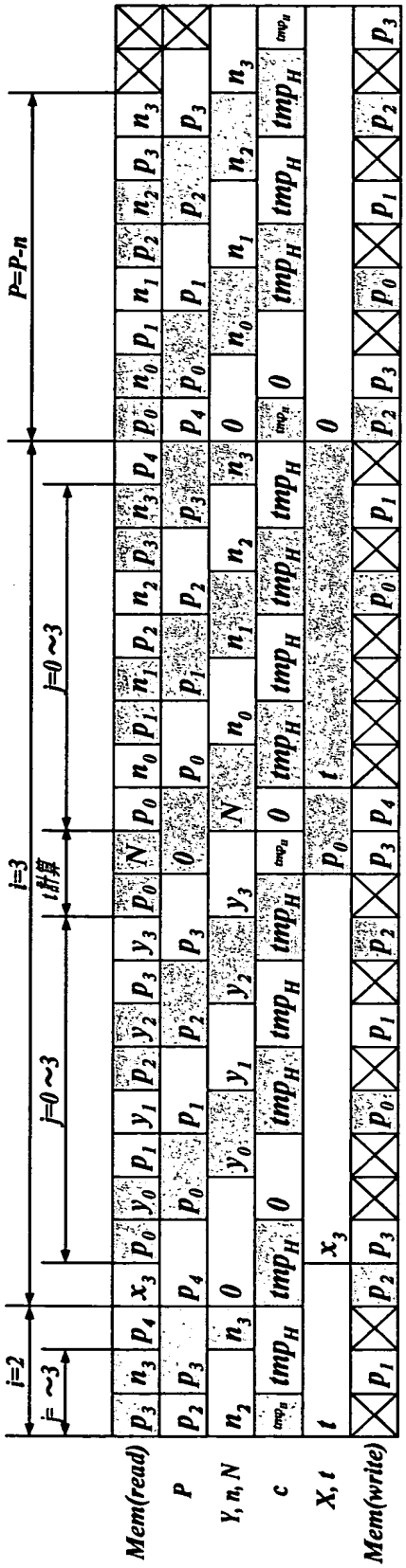
【図６】


Mem A(read)
Mem B1(read)
Mem B2(read)
P
c
t
n,N
X
Y
Mem A(write)
Mem B2(write)
i=2
j= ～3
t計算
i=3
j=0 ～3
P=P-n
Mem A → MemB

【図７】


tmp
2r
c
p_j
$y_j \cdot x_{i(0)}$
$y_j \cdot x_{i(1)}$
$y_j \cdot x_{i(r-1)}$
+
tmp_L
tmp_H
r
c
x_i, t
p_0
y_j, n_j
N
p_j
tmp_H
2:1
tmp_L
t
x
メモリ
P,X,Y,n,N
$tmp \leftarrow p_j + x_i \cdot y_j + c$
$tmp \leftarrow p_j + t \cdot n_j + c$
$tmp \leftarrow p_0 \cdot N$

【図８】



【書類名】　　要約書

【要約】

【課題】　モンゴメリ乗算回路におけるメモリアクセスのボトルネックを２ポートやシングルポートの汎用メモリを使いながら解消する。

【解決手段】　２つのメモリ７、８を設け、演算に必要な変数のうちメモリから読み出す必要のある変数については異なるメモリに記録する。そしてパイプライン処理の同一の読出しステージでメモリ７からレジスタ５に変数を読出し、メモリ８からその他のレジスタに他の変数を読み出す。

【選択図】　図１

認定・付加情報

特許出願の番号	特願２０００－３８６０６９
受付番号	５００００１６３９４０４
書類名	特許願
担当官	濱谷　よし子　　１６１４
作成日	平成１３年　２月　８日

＜認定情報・付加情報＞

【特許出願人】
- 【識別番号】 390009531
- 【住所又は居所】 アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　（番地なし）
- 【氏名又は名称】 インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】
- 【識別番号】 100086243
- 【住所又は居所】 神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内
- 【氏名又は名称】 坂口　博

【代理人】
- 【識別番号】 100091568
- 【住所又は居所】 神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内
- 【氏名又は名称】 市位　嘉宏

【代理人】
- 【識別番号】 100106699
- 【住所又は居所】 神奈川県大和市下鶴間１６２３番１４　日本アイ・ビー・エム株式会社大和事業所内
- 【氏名又は名称】 渡部　弘道

【復代理人】 申請人
- 【識別番号】 100112520
- 【住所又は居所】 神奈川県大和市中央林間３丁目４番４号　サクライビル４階　間山・林合同技術特許事務所
- 【氏名又は名称】 林　茂則

【選任した復代理人】
- 【識別番号】 100110607

次頁有

認定・付加情報（続き）

【住所又は居所】　神奈川県大和市中央林間３丁目４番４号　サクライビル４階　間山・林合同技術特許事務所
【氏名又は名称】　間山　進也

次頁無

出　願　人　履　歴　情　報

識別番号　　〔３９０００９５３１〕

１．変更年月日　　２０００年　５月１６日
［変更理由］　名称変更
住　所　アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　（番地なし）
氏　名　インターナショナル・ビジネス・マシーンズ・コーポレーション